Exploring Parallelization for Medium Access Schemes on Many-core Software Defined Radio Architecture

Xi Zhang, Junaid Ansari, Manish Arya and Petri Mähönen
RWTH Aachen University, Institute for Networked Systems,
Kackertstrasse 9, D-52072 Aachen, Germany
xzh@inets.rwth-aachen.de

ABSTRACT

As multi-standard devices and high speed communication standards are emerging, timeliness requirements and flexibility for both baseband modem and medium access schemes are becoming essential. Software Defined Radios (SDRs), in this context, aim at offering the desired flexibility while satisfying the real-time constraints. An SDR architecture consisting of many-core homogeneous computing elements provides easy protocol implementation, a high level of portability and extension possibilities. It does not require architecture specific program code which is needed by the popular heterogeneous SDR architectures. Therefore, in this paper, we explore how a homogeneous SDR architecture is used for efficient realization and execution of Medium Access Control (MAC) protocols. In particular, we investigate the performance of two broad classes of MAC schemes on the Platform 2012 (P2012) many-core programmable computing fabric. We provide a toolchain which utilizes the characteristics of P2012 for MAC parallelization, runtime scheduling, and execution. Our results indicate that by using the supporting toolchain, reconfigurable MAC implementations are able to exploit the computational power offered by the platform and adhere to the timeliness constraints. Computationally intensive algorithms for MAC layer parameter optimization show an improvement of up to 85% in the convergence time as compared to using a single-core architecture.

Categories and Subject Descriptors
C.2.1 [Network Architecture and Design]: Wireless communication

General Terms
Design, Experimentation, Performance

Keywords
MAC, Parallelization, Many-core, SDR platform

1. INTRODUCTION

MAC procedures for emerging network technologies demand fine grained timing control, high computational power and adaptability. MAC layers for high data rate standards such as LTE and IEEE 802.11ac require fast real-time response. As spectral resources are shared by multiple networks, different spectrum sharing policies are enforced which require flexibility at the MAC layer. In the context of Cognitive Radios (CRs) paradigm, often machine learning techniques and statistical data analysis algorithms are employed for runtime resource management, channel allocation at MAC layer, and performance optimization based on the PHY- and MAC parameters. Machine learning algorithms are typically computationally intensive and a short convergence time is desired for rapid runtime adaptation of MAC protocols.

SDR platforms emerge to be an attractive option for fulfilling high flexibility demands. These platforms are typically equipped with computationally powerful units such as GPPs to handle the baseband modem signal processing while offering a high level of flexibility [1,2]. Performance results, however, indicate that this approach has shortcomings in meeting strict timeliness and scheduling requirements of MAC processing [3]. In this architecture, the MAC layer is typically isolated and the need for fast PHY/MAC interaction is not satisfied. Coprocessor-FPGA architectures have shown to outperform the GPP based protocol processing units [4]. In order to efficiently meet the computational load of evolving high data rate standards, multiprocessor architectures are proposed to achieve parallelization gains for baseband signal processing. The current multiprocessor architecture trend is shifting from multi-core to many-core. Ron Wilson has pointed out that there is a shift in the concept from making one processor powerful enough to handle all the data towards a more distributed approach of many processors sharing the work [5]. While heterogeneous architectures are commonly used and are known for their power efficiency, flexibility and reconfigurability offered by them remain limited as large implementation and debugging efforts are needed whenever extensions or modifications are made [6]. Homogeneous architecture, on the other hand, makes it easier for a software designer to map the applications, threads, parallel tasks, onto any processors in a flexible way as no considerations for different types of processors and hardware accelerators are necessary. The required implementation efforts and the resulting performance characteristics also heavily depend upon the tools and software support provided by an SDR platform. Highly dynamic MAC protocols, especially for CR environments, require specialized tools and frameworks to efficiently schedule tasks and manage radio resources at runtime in order to achieve desired reconfiguration and adaptation.

In this paper, we describe the implementation details and performance characteristics of two main types of MAC procedures: classical CSMA based MAC protocols and reconfigurable MAC schemes using machine learning based algorithms for runtime optimization and resource management. The implementation is carried out on a cycle accurate emulator of a homogeneous computing fabric P2012 [7]. P2012 is a power-efficient many-core comput-

SRIF'13, August 12, 2013, Hong Kong, China.

ing fabric, which is based on four globally asynchronous, locally synchronous (GALS) clusters and is developed in 28nm CMOS technology. P2012 has been used to realize the inner modem of a MIMO OFDM transceiver with maximum 4 x 4 antenna configuration and timing requirements from IEEE 802.11n standard [8]. Although the primary target of the platform is not baseband processing, real-time performance has been achieved using 8 Processing Elements (PEs) benefiting from the hardware synchronizer and shared memory architecture offered by the platform. We have realized a toolchain for runtime protocol realization TRUMP [9] on P2012 for parallelization and scheduling of MAC processes. We explore how a MAC layer protocol benefits from the supported tools and capabilities of P2012 with homogeneous computing elements.

The rest of the paper is outlined as follows. In Section 2 we review different platform architectures used by the SDR community and how they perform in terms of flexibility, scalability, and programmability. We introduce P2012 in Section 3 together with its software tools support. Section 4 presents the evaluation results of our two main categories of MAC protocol implementations. Finally, we conclude the paper in Section 5.

2. SDR PLATFORM ARCHITECTURES

Various technologies such as ASICs, FPGAs, DSPs, and GPPs, have been used in SDR platforms, commercially available products, military applications, and research prototypes. These technologies are used in both standalone and hybrid fashion [10]. In this section, we discuss the characteristics of different technologies and their implications on real-time MAC protocol performance and their suitability for emerging MAC protocols which demands high level of flexibility and computational power.

2.1 Task Specific Processing Elements

In standardized network interface cards such as IEEE 802.11 b/g where flexibility and reconfigurability requirements are limited, an ASIC is often used to handle the physical and MAC layer processes. ASIC solutions are highly optimized for realizing a particular computationally demanding protocol algorithms. They offer high level of computational efficiency and low power consumption. Although ASIC implementations are static and rigid, they can be suitable for implementing common functionalities across different configurations to accelerate the protocol execution speed and lower the power consumption [11]. The idea of implementing common and computational intensive functionalities in hardware for speed gains instead of pure software has also been proposed [12]. While many vendors provide standard compliant NICs, Bianchi *et al.* have shown that having programmability and reconfigurability at MAC layer helps in increasing the achieved throughput [13]. ASIPs are typically tailored to a specific application and exhibit a lower energy consumption than GPPs or DSPs while offering more flexibility than ASICs [14]. However, new standards demanding high flexibility, reconfigurability and multi-mode operation, make both ASICs and ASIPs alone not a very viable option for SDR platform implementation.

2.2 Reconfigurable Processing Elements

As compared to ASICs and ASIPs, FPGA is a reconfigurable solution at the expense of lower processing speed, higher power consumption and circuit area. There are several SDR development platform implemented based on FPGA. As an example, WARP boards [15] developed by Rice university are built using Xilinx Virtex FPGA and aim at offering flexibly PHY/MAC layer development. Computational intensive processes, signal processing are implemented in the FPGA while the application layer and some control functionalities are implemented in the PowerPC core in the FPGA. WiNC2R [16] is another example of a SDR platform built on a FPGA with soft-core processors and accelerators. Runtime reconfiguration of FPGA can be realized by partial reconfiguration, which requires significant efforts in FPGA development and is highly dependent on the tools and devices available. It can also be realized by software programmable reconfiguration, which has the limitation that all the program component has to be implemented before hand and the configuration options are limited to the controlling parameters which have been exposed to the soft-core. FPGA based architecture is more suitable for experimentation and prototyping than standardized commercially available SDR platforms due to the relatively slow processing speed. Furthermore, since the size of FPGAs is limited, it does not offer good scalability and is expensive to implement multiple computational intensive algorithms for MAC layer optimizations

2.3 General Purpose Processors

Microprocessor systems provide full real-time programmability [17]. CalRadio [18] is a flexible wireless platform developed at UC San Diego targeting at fully programmable MAC protocols. It uses Intersil HFA3836 baseband chip for IEEE 802.11b PHY layer implementation which offers parameters such as the data rate and transmit power to be controlled by the MAC layer through register configurations. CalRadio provides a DSP for MAC layer implementation entirely in software which allows a high degree of flexibility in MAC layer design, though the packet transfer delay from host to DSP to PHY has limited the throughput to IEEE 802.11b PHY layer [19]. GPPs are typically unable to handle wideband signal processing in a timely manner to comply with the protocol standard. Therefore, multi-core architectures are introduced to achieve better performance by parallelizing processes. Parallel operations significantly reduce the execution speed and the power consumption per instruction. Sora [20] exploits parallelism in the MAC/PHY layer processing and is able to comply with IEEE 802.11 b/g standard. However, modification and extension to MAC/PHY implementations on Sora are highly complicated due to the sophisticated distribution of computational processes on multi-core processors in an effort to meet the real-time requirements. USRP1.0 [1] does the baseband signal processing are done on the host PC implemented using GNU Radio or NI LabView. The throughput achieved on USRP boards is typically little since the CPU processing power is the bottleneck. GPP based approach is good for fast PHY/MAC layer development. However, the processing latency and the power consumption are two major issues. Therefore, GPPs and/or DSPs often require hardware acceleration. Lau *et al.* have discussed the use of FPGA and ASIP based hardware accelerator in SDR waveforms and concluded that hardware accelerator enhances power efficiency which is essential in making SDR platforms into mobile terminals and handsets [21].

2.4 MPSoC Approach

Multiprocessor System-on-Chip (MPSoC) consists of multiple programmable processors. Heterogeneous multi-core architecture is popular for its power efficiency, high performance and low cost. IMEC's baseband engine for adaptive radio (BEAR) platform consists of six cores (three ASIPs, one ARM processor, two architecture for dynamically reconfigurable embedded systems (ADRES) processors) and two accelerators [22]. Infinion MuSIC-1 platform [23] is also a heterogeneous multi-core platform which consists of four programmable DSP cores and accelerators for FIR filter, Viterbi decoder, etc. The heterogeneity allows different processes

to be implemented on the most appropriate processor and thus achieve a speed-efficient solution. MAGALI platform [24] is a heterogeneous Network-on-Chip (NoC) based MPSoC platform dedicated for mobile terminals. This platform uses a centralized control processor for achieving power efficiency. However, this scheme limits the scalability of the architecture. The limitation applies to heterogeneous architecture in general. Moreover, as the complexity increases in heterogeneous architecture, often with irregular organization of memory hierarchy, efficient mapping of protocol algorithms on them is difficult [25].

Homogeneous many-core architecture provides a mid-way between multi-core CPUs and Graphics Processing Units (GPUs) for a balance between programmability and parallelism. GENEPY (homoGENEous Processor ArraY) platform [6] is purely homogeneous, with Smart ModEm Processors interconnected with a NoC. Although homogeneous architecture is, in general, believed to be less efficient in speed and power consumption than heterogeneous architecture at the expense for offering easier programmability, higher flexibility and scalability, the authors have shown that for an LTE application, GENEPY has performance gains of 3% in speed and 18% in power consumption as compared to MAGALI platform. In this paper, we have used P2012 which consists of homogeneous processing clusters for MAC protocol parallelization.

3. P2012 ARCHITECTURE

In order to span the wide efficiency spectrum between fully programmable homogeneous many-cores and application specific accelerators, a new family of computing systems called Many-Core Computing Fabrics (MCCFs) have been introduced. MCCF consists of many homogeneous processing cores interconnected by a NoC infrastructure [26]. P2012 is an area- and power-efficient MCCF developed by ST Microelectronics. It aims at filling the gap between general-purpose embedded CPUs and fully hardwired application accelerators in terms of area and power efficiency. It is flexible in supporting a wide range applications while not losing power efficiency. P2012 is based on four GALS clusters. One cluster consists of a multi-core computing engine called ENCore and a Cluster Controller (CC). The EnCore cluster can host up to 16 PEs. All EnCore PEs share a L1 tightly coupled data memory (TCDM) which supports a throughput one data access per PE per clock cycle. A low latency network is used to interconnect the PEs and on-chip shared memory banks within an EnCore cluster. A hardware synchronizer is used for ENCore to provide scheduling and synchronization acceleration. The hardware synchronizer also has a dynamic allocator which allows the system to dynamically allocate the best available PE to execute a task, which is suiting to the flexible and unpredictable nature of MAC processes in a dynamic environment. The CC takes care of booting and initializing the ENCore PEs and deploying applications onto the PEs. It consists of a Direct Memroy Access (DMA) subsystem which transfers data blocks between the external memory and the internal memory during operation of PEs. The fast memory access facilitates MAC protocol realization especially in meeting the real-time requirements. Furthermore, the memory among different clusters are transparent on the platform, i.e. one cluster can directly access the memory on other clusters. No memory copying overhead is induced and therefore real-time MAC execution can be realized as MAC processes typically involve multiple data accesses with low-delay tolerances. These clusters are connected via a high-performance fully-asynchronous NoC. The clusters are implemented with independent power and clock domains, enabling aggressive fine-grained power, reliability and variability management.

3.1 Software Tools

There are mainly two layers in the software stack for P2012. The runtime layer interacts directly with the P2012 fabric and provides basic functionalities such as task scheduling, dispatching, memory allocation, resource and power monitoring, host-fabric communication, etc., to upper layers. The programming model layer provides high level environment for developing specific programming models and applications. P2012 supports industrial standard programming models such as OpenCL and OpenMP programming models. We have developed our MAC layer schemes for P2012 using the Native Programming Model (NPM). The NPM allows developing specific applications running on a P2012 fabric and integrating them with the host system. The NPM is highly optimized for the P2012 architecture. It takes into account the specific features of the P2012 architecture like direct access to hardware synchronizer and DMA, or the partition between CC and ENCore Processors, thus providing the highest level of control on application-to-resource mapping at the expense of abstraction. Our MAC schemes fully utilize the NPM capability to achieve fast execution and provide fast response to the network. Applications for P2012 require execution engines to manage the interaction between CC and ENCore processors. Execution engines provide methods for initializing, starting, notifying and stopping ENCore PEs. The Reactive Task Manager (RTM) is an execution engine supported by NPM. RTM runs in a cluster and allows easy fork/join and duplication of jobs on PEs. The Multi-Thread Engine (MTE) is another execution engine available through NPM. The MTE uses threads to parallelize processes. Barriers are used for synchronization and the threads cannot be preempted. Multiple threads can run on either single or multiple PEs within one cluster. Using both MTE and RTM based on their different capabilities in mapping tasks to PEs, we have implemented a toolchain TRUMP for MAC protocol parallelization and scheduling.

3.2 TRUMP

TRUMP is a toolchain for runtime protocol realization. It consists of a MAC meta descriptor for MAC protocol design in C-like syntax, MAC meta compiler which interprets the MAC description for the target platforms, and Wiring Engine for managing the runtime execution of the MAC protocol. TRUMP aims at providing parallelization possibilities of independent MAC processes. It has a dependency table which captures the dependencies among different MAC processes, and a logic controller which governs the scheduling of the MAC processes based on the availability of thread/processor core and the state machine of the MAC protocol. We have used TRUMP on a x86 Linux based multi-core PC for simulation of some MAC protocol configurations. A reduction in terms of execution speed of 90% is observed in our test case as compared to a single core single thread environment. We have implemented TRUMP on to P2012 using NPM and its execution engines for easy MAC protocol realization as shown in Figure 1. As part of TRUMP, the MAC meta-compiler is implemented on the host side which processes both the MAC description and the dependencies indicated by the MAC designer for the functions. An execution list which contains the logic and functions used by the MAC description is formed. The list is analyzed and a two-dimensional array is written with the logic operator, functions and the dependency code associated. The array is passed to the Fabric Controller (FC) as an argument of a message. The FC uses the Wiring Engine to map the functions in the array to the MAC components in the library. Depending on the nature of the MAC layer applications, TRUMP uses different runtime execution engines. MTE and RTM can also be used together on different clusters. For example, stan-

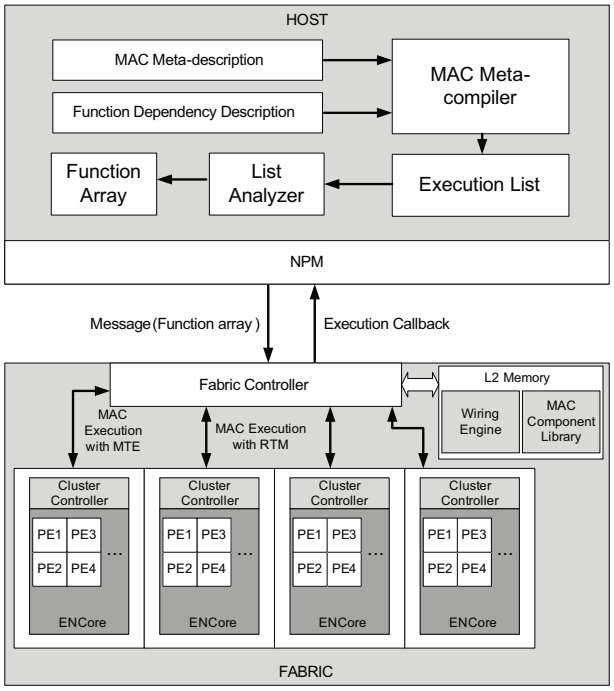

Figure 1: System architecture: TRUMP implementation on P2012.

dard MAC protocol processes uses MTE since individual thread terminates independently. RTM is more suitable for parallelizing of duplicated tasks and callback function is only triggered when all the tasks on the PEs within one cluster are done execution, which is suitable for some of the machine learning algorithms for MAC schemes.

4. EVALUATION RESULTS

In order to assess the benefit and drawback that many-core architecture brings for new generation of MAC protocols, we have implemented three types of MAC layer applications on a cycle accurate P2012 emulator for evaluation: a) classical MAC schemes which do not have significantly computational algorithms; b) genetic algorithm based runtime MAC performance optimization algorithm, and c) swarm intelligence based channel selection algorithm. Additionally, since multi-core architecture requires additional scheduling and management mechanisms, we also present the execution overhead for task scheduling and the initialization and termination overheads of the platform. The error rate of the cycle-accurate emulator is around 10%.

4.1 Classical MAC Executions

We have implemented a simple MAC protocol as shown in Table 1 using TRUMP. Since the P2012 does not have a radio front-end, we have used the timing measurements from WARP SDR boards for all the functions that are used in this protocol to emulate a more realistic behaviour. Some of the functions are independent from others and can be executed in parallel, e.g., `BackOff()` and `SetFrequencyChannel()` while some functions have to be executed in a specific sequence, e.g., `SendPacket(ACK)` has to be executed after `WriteToTxBuffer(ACK)`, as the transmit buffer needs to be filled with the relevant data before a transmission should take place. We have measured the complete execution

Table 1: MAC protocol description and the measured execution duration associated with the MAC functions from WARP platform.

MAC Protocol Description	*Duration [us]*
`label Start;`	
`WriteToTxBuffer(DATA);`	29.5
`label TryToSend;`	
`BackOff();`	18
`SetFrequencyChannel();`	22
`if(CarrierSensing())`	30
`  SendPacket(DATA);`	433
`  if(WaitForPkt(ACK))`	144
`  ReadFromRxBuffer();`	20
`  goto Start;`	
`  else`	
`  goto TryToSend;`	
`  endif`	
`else`	
`  if(WaitForPkt(DATA))`	433
`  ReadFromRxBuffer();`	50
`  WriteToTxBuffer(ACK);`	11
`  SendPacket(ACK);`	144
`  endif`	
`  goto TryToSend;`	
`endif`	

time of the above described MAC protocol on P2012 using different number of PEs. Figure 2 shows the ratio of the execution time of parallelized MAC protocol using multiple PEs against sequential execution using one PE over varying number of iterations. Approximately only 3% of execution time has been saved by using parallelization for this MAC protocol realization. It is due to the high level of dependency of the function used in this protocol and the short execution time required by each functions.

We have analyzed the complete execution time and identified two main parts contributing to the overhead of executing a MAC protocol on the P2012. The initialization overhead includes the initial communication between the host and the fabric and the initialization of fabric. The scheduling overhead refers to the time taken for the CC to schedule tasks onto the PEs at runtime. There is always a delay between task executions due to the central controller. Even for independent MAC processes, there is a difference in the starting time since they need to be allocated onto the PEs in sequence. Figure 3 shows the initialization and scheduling overhead of executing a MAC protocol. It can be seen that the overheads of using two PEs and three PEs are almost the same while the one PE results in a significant increase in the scheduling overhead. It is mainly due

Figure 2: The ratio of the execution time of the MAC protocol described in Table 1 using multiple PEs for parallelization against sequential execution on one PE.

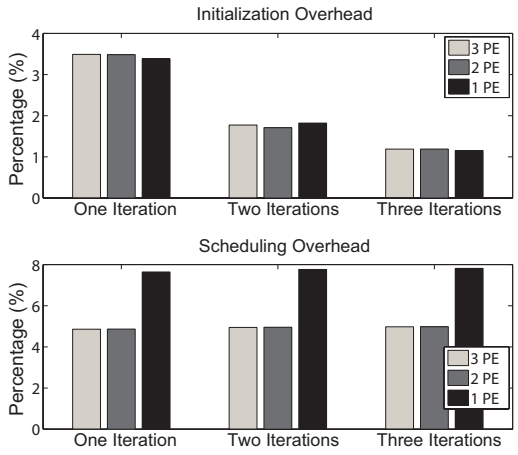

Figure 3: The execution overhead of the MAC protocol described in Table 1 on different numbers of PEs in one cluster with different numbers of iterations.

to the shortened total execution time of the program by being able to parallelize MAC processes onto multiple PEs. The initialization overhead is around 3% for one iteration of the MAC protocol. Since MAC protocol runs a long time, the initialization overhead is almost negligible. There is no significant difference among the initialization overhead in this test case since the number of PEs used is small. We show the initialization overhead of the whole P2012 fabric in Section 4.2.

4.2 Genetic Algorithm based MAC Parameter Optimization

Genetic Algorithms (GAs) have been widely used in the MAC research community for parameter optimization and performance adaptations. GAs [27] are a method of search, mainly for learning and optimization purposes. Although GA is typically not fast to converge, it is robust, scalable and well suited for optimization problems involving large search spaces. The GA computation starts from definition of fitness functions which are measures of performance towards an or multiple objectives. A few randomly generated populations of individuals known as chromosomes are selected. Each chromosome represents a possible solution to a problem. The fitness of each chromosome in a population in each generation is evaluated based on the fitness functions. Crossover and mutation are actions performed to the chromosomes of the populations at each generation. In general, chromosomes which are believed to be able to survive from generation to generation are selected and the population of the new generation is expected to be better than the old generation. The process is iterated until convergence criteria are met [28].

GA has been used for an engine which provides awareness-proce ssing, decision-making and learning elements of cognitive functionality [29, 30]. Since CRs are required to adapt based on the environmental sensing and learning results, GAs are used to evolve a radio defined by a chromosome. The adjustable parameters of a radio are presented by genes in a chromosome and a set of parameters which optimizes the radio for the CR node needs is found by GA. GA has also been used in other aspects for CRs such as spectrum management [28], determining of Radio Frequency (RF) parameters for optimal radio communications in the varying RF environment for autonomous vehicle communications [31], and channel allocation [32]. Although GA has been proposed to be used in various areas for Cognitive Radio Network (CRN), the high computational requirement of the algorithm has been a hurdle to realize GA in real-time for resource optimization in the MAC layer [33].

Since it is an inherent nature of GAs that the evaluation of individuals can be conducted independently, the computing time can be effectively accelerated by means of parallel computation [34]. Parallel GAs also have the advantage of modeling natural evolution more closely by introducing the concept of spatial locality [35]. Therefore, we see great potential in employing GA for MAC layer optimization at runtime by exploiting the parallelism on many-core architectures.

4.2.1 Problem Formulation

In this work, we implement a GA with multi-objectives. Our implementation is adapted based on previous work from Newman *et al.* [33]. We have included a list of PHY and MAC parameters as the genes for the chromosomes. The genes include radio transmission power, modulation type, modulation index, frequency channel, number of subcarriers, channel coding rate and packet size. A population of 100 chromosomes each of length of 44 bits are generated randomly. The crossover rate is set to be 90% and mutation rate is 5%. The fitness function is defined as

$$f = w_1 * (f_{\text{min\_power}}) + w_2 * (f_{\text{min\_per}}) + w_3 * (f_{\text{max\_throughput}}), \quad (1)$$

where

$$f_{\text{min\_per}} = 1 - \frac{\log 0.5}{\log P_{\text{pe}}}, \quad (2)$$

and

$$P_{\text{pe}} = 1 - (1 - P_{\text{be}})^L. \quad (3)$$

The definitions of $f_{\text{min\_power}}$, $f_{\text{max\_throughput}}$ and P_{be} are according to [33] while $f_{\text{min\_per}}$ refers to the minimum packet error rate P_{pe}, which is related to the bit error rate P_{be} and the length of the packet L. w_1, w_2, and w_3 are the weights assigned to each individual fitness functions depending on the application scenario and user requirements.

Since the fitness function execution is computationally intensive, involving complicated mathematical functions, we have parallelized the execution of fitness function calculation across chromosomes. Each chromosome is assigned to a PE for the fitness function calculation while the rest of the processes such as mutation, crossover, chromosome selection and replacement is done at the host.

4.2.2 Execution Time

In this experiment, we execute the GA using varying number of clusters and PEs to analyze the benefit and overhead in terms of execution time. Figure 4 shows the overhead of using multiple clusters and PEs on P2012 for executing the GA. The initialization overhead increases significantly with the number of clusters while the termination overhead is comparatively negligible. These overheads just occurs once, i.e. when a node is booted and shut-down. Therefore, they are insignificantly small (less than 80 ms) when the lifetime of a node can be of days and months.

Figure 5 shows the actual execution time (excluding the initialization and closing overhead) of the GA on P2012 fabric. We can see that with small number of generations, the execution time does not differ too much with different amount of computational power, especially with multiple clusters. However, a GA easily takes approximately 200 iterations to reach a reasonable fitness score. Significant improvement is shown when using more clusters and more

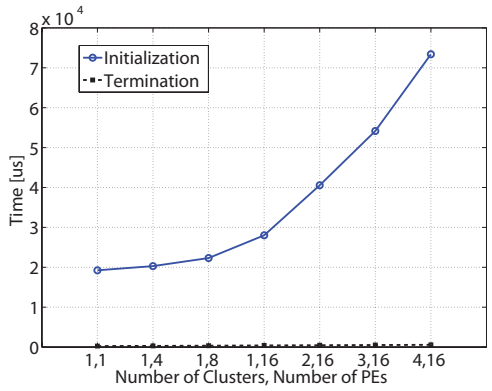

Figure 4: The initialization and termination overhead of using different number of PEs and clusters of the P2012 fabric when executing the parallel GA.

PEs as the number of generation increases. In an extreme case, an 85% improvement is observed for 200 iterations when four clusters with 64 PEs in total are used as compared to one PE. It shows that when a change has occurred in terms of either application QoS requirements, or special condition, or network topology, etc., which demands an adaptation at the MAC scheme, the node is able to find an acceptable solution in 10 ms if equipped with the whole P2012 fabric for parallel execution as compared to 75 ms with sequential execution.

4.3 Swarm Intelligence Algorithm for MAC Layer Channel Allocation

Inspired by the observation that social insects such as ants and bees work in a self-organized fashion with unsupervised coordination between simple interactions among individuals in the colony, Swarm Intelligence (SI) algorithms model network users as a population of simple agents interacting with the surrounding environment [36,37]. Although each agent has little intelligence, global intelligence is resulted from the collaborative behaviour of the colony. Division of labour is the key in SI where different activities are performed by those who are better suited to the task. SI algorithms have three characteristics which made them popular in a wide range of applications: flexibility in adapting to a changing environment, robustness against failure of individuals, and self-organization with unsupervised activities [38]. Although in this work we have applied SI for MAC layer channel allocation, SI has been widely popular in resource management in CRNs in general. Doerr *et al.* have used SI algorithm to dynamically identify common control channel in CRNs [39]. SI algorithms have also been used for optimum resource allocation in terms of assigning available spectrum holes to CR users [40, 41]. BIOlogically-inspired Spectrum Sharing (BIOSS) algorithm allocates channels to unlicensed users in CRNs based on the adaptive task allocation model in insect colonies. BIOSS works with distributed network architecture since the users can distributively select the channels for communication [42]. BIOSS has been enhanced in channel allocation so that channels which have the minimum excess power over the node's transmission power is preferred [43]. This modification has improved the utilization of low-power channels and thus the global spectrum utilization.

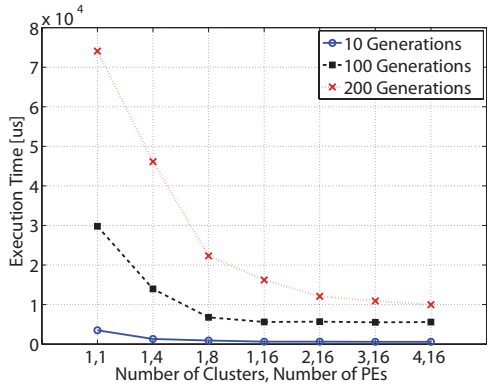

Figure 5: The execution time of the GA with different number of generations using different number of PEs and clusters of the P2012 fabric.

4.3.1 Problem Formulation

We have followed the enhanced BIOSS protocol (eBIOSS) [43] closely for our channel allocation algorithm implementation. A brief overview of the algorithm is described here. The probability of selecting a particular channel which satisfies the transmission requirement is

$$T_{ij}^{csp} = \begin{cases} 0, & P_j < p_i \\ 1 - \frac{P_j^n}{P_j^n + \alpha p_{ij}^n}, & P_j \geq p_i \end{cases} \quad (4)$$

where P_j is the permissible power to channel j, p_{ij} is the required transmission power of node i to channel j. It is assumed that the permissible power to all channels are available at each node through spectrum sensing, and the required transmission power can be determined according to the user requirements and channel characteristics. n determines the slope of the channel selection probability T_{ij}^{csp} and is set to be 2 in our implementation. α is a constant which determines the influence of p_{ij} and is set to be 10.

The channel selection probability is calculated for all the channels at a node when an event happens such as a transmission task arises, the operating environment changes, the QoS requirement varies, etc. The channel with the maximum T_{ij}^{csp} is selected. The selection is then evaluated and good channels are remembered.

4.3.2 Channel Allocation

In our experiment, we assume a one-hop network with fixed number of channels and nodes contending for transmission. Each node is assigned with a required transmission power per transmission task randomly. Each node is equipped with a P2012 fabric. We initialize each node with a random number of clusters (1 to 4), which helps us to evaluate the relationship between the processing capability and the opportunity in finding a suitable channel for transmission. Since there is typically more demand than supply, i.e. more nodes within a network than the channels which satisfy the required transmission power, some nodes might not find a suitable channel every time. Channel algorithms are performed repetitively until a channel is allocated for the pending transmission task. Figure 6 shows the possibility for a node to find a suitable channel in a network of varying size. We see that the nodes which only have one cluster for executing eBIOSS algorithm performs the worst while the rest nodes have similar level of performance. It is due to the fact that since the number of total available channel is 20 and the number of PEs in one cluster is only 16, it takes much longer for the nodes with one cluster to calculate the channel selection probability for all the channels. Since channels are allocated

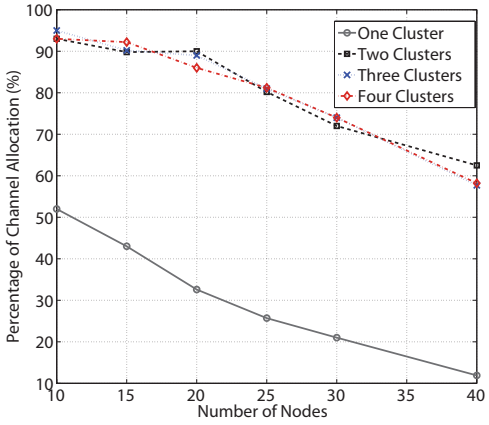

Figure 6: The channel allocation rate of varying sizes of network with different number of clusters for executing the channel allocation algorithm. The total number of channels is 20.

based on a first come first serve basis, the nodes which are able to process information faster have higher chance in grabbing the channels which fit to their transmission requirement. This result also shows the relationship between network parameters and the amount of computational power required. The performance curves of two, three and four clusters lie very close together. It shows that clusters more than two are not required for this application scenario. Involving more than necessary processing elements results in extra control overhead. Figure 7 shows the possibility to obtain a suitable in a network with different number of channels available. The more channels available, the more computational power is required from the node to be able to evaluate the channel quality on time. We see that when only 10 channels are available, nodes with more clusters do not have any advantage over nodes with only one cluster and the successful channel allocation rates are the same for all the nodes. When the number of channels increases, high computational power becomes more beneficial.

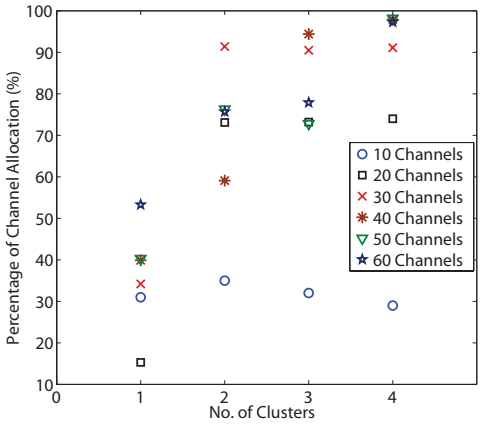

Figure 7: The channel allocation rate at nodes with different computational power of a network with 30 nodes. The number of channels varies from 10 to 60.

5. CONCLUSIONS

With increasing requirements of hard real-time constraints as well as high degree of flexibility for MAC-layer algorithms, SDR community is exploring multi-core architectures. Since heterogeneous platforms exhibit restricted possibilities for extension and compatibility of the legacy code, we have investigated MAC-layer realization on the homogeneous P2012 architecture. An OFDM MIMO PHY layer implementation has already been realized for P2012 [8]. At the MAC-layer, we focus on the efficient parallelization and scheduling of MAC processes benefiting from the platform architecture and its software tools. We use our toolchain TRUMP to schedule different MAC processes on different computing elements according to the state machine of a particular MAC scheme and the hardware resource availability at runtime. For evaluation, we have considered two main classes of MAC scheme realizations. We have observed that classical simple CSMA/CA based MAC protocols benefit little from parallelization. This is owing to the fact that these protocols do not involve heavy computations and the MAC processes exhibit short execution and blocking times. Although the execution speed gain is only 3%, it shows that the scheduling overhead and communication delays induced for a many-core architecture can be compensated by parallelization in simple MAC protocols. For MAC protocols involving complicated algorithms, many-core platform architectures show significant benefits from parallel execution capability. Advanced channel selection and resource management schemes often require machine learning based or statistical data analysis methods. Due to the tight scheduling and timeliness constraints of reconfigurable MAC schemes, these computationally demanding algorithms cannot be offloaded externally due to data and control bottlenecks. Therefore, the on-board processing power offered by many-core architecture is desired. We have shown that by fully exploiting the computational power on P2012, we are able to achieve an up to 85% improvement in convergence time when using genetic algorithm for MAC/PHY parameter optimization as compared to using a single-core platform. We have also shown that when using SI algorithm for channel allocation, it is 2-6 times more likely for a node with more computing power to get a desirable channel than a node with limited computational power. We believe that our results show the importance of parallelization of computationally complex MAC strategies and emphasize the need for many-core architecture in SDR platforms.

Acknowledgment

We thank partial financial support from RWTH Aachen University, DFG through UMIC research centre (Nucleus project) and EU through 2PARMA (FP7-248716) project.

6. REFERENCES

[1] "Resources for GNU Radio and USRP boards." `http://www.ettus.com/resource` [Last visited: 1st Oct, 2012].

[2] "Software Radio Laboratory LLC Wiki." `http://qs1r.wikispaces.com/` [Last visited: 1st May, 2013].

[3] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste, "Enabling MAC Protocol Implementations on Software-Defined Radios," in *Proceedings of the Symposium on Networked Systems Design and Implementation*, (Boston, MA, USA), 2009.

[4] H. Lee and T. Mudge, "A Dual-Processor Solution for the MAC Layer of a Software Defined Radio Terminal," in *Proceedings of the ACM International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, (San Francisco, CA, USA), 2005.

[5] R. Wilson, "From Multicore to Many-Core: Architectures and Lessons," *System Design in the 28-nm Era*, October 2012. `http://www.altera.com/technology/system-design/articles/2012/multicore-many-core.html`.

[6] C. Jalier *et al.*, "Heterogeneous vs Homogeneous MPSoC Approaches for a Mobile LTE Modem," in *Proceedings of Design, Automation and Test in Europe*, (Dresden, Germany), 2010.

[7] L. Benini, E. Flamand, D. Fuin, and D. Melpignano, "P2012: Building an Ecosystem for a Scalable, Modular and high-efficiency Embedded Computing Accelerator," in *Proceedings of the Conference and Exhibition on Design, Automation and Test in Europe*, (Dresden, Germany), 2012.

[8] D. guenther, T. Kempf, A. Ishaque, and G. Ascheid, "Systematic MIMO OFDM transceiver implementation for MPSoCs: a necleus based approach," *Analog Integrated Circuits and Signal Processing*, vol. 73, pp. 597–612, 2012.

[9] X. Zhang, J. Ansari, G. Yang and P. Mähönen, "TRUMP: Supporting Efficient Realization of Protocols for Cognitive Radio Networks," in *Proceedings of the IEEE Symposium on New Frontiers in Dynamic Spectrum*, (Aachen, Germany), 2011.

[10] M. Dardaillon, K. Marquet, T. Risset, and A. Scherrer, "Software Defined Radio Architecture Survey for Cognitive Testbeds," in *Proceedings of the International Wireless Communications and Mobile Computing Conference*, (Cyprus), 2012.

[11] J. L. Shanton III and H. Wang, "Design considerations for size, weight and power (SWAP) constrained radios," in *Proceedings of the Software Defined Radio Technical Conference and Product Exposition*, (Orlando, FL, USA), 2006.

[12] V. Ramakrishnan *et al.*, "Efficient and portable SDR waveform development: The Nucleus concept," in *Proceedings of the Military Communications Converence*, (Boston, USA), 2009.

[13] G. Bianchi *et al.*, "MAClets: active MAC protocols over hard-coded devices," in *Proceedings of the 8th International Conference on emerging Networking Experiments and Technologies*, (Nice, France), pp. 229–240, 2012.

[14] C. Schmidt-Knorreck *et al.*, "Flexible Front-End Processing for Software Defined Radio Applications Using Application Specific Instruction-Set Processors," in *Proceedings of the Conference on Design and Architecture for Signal and Image Processing*, (Cagliari, Italy), 2012.

[15] A. Khattab, J. Camp, C. Hunter, P. Murphy, A. Sabharwal, and E. W. Knightly, "WARP: a flexible platform for clean-slate wireless medium access protocol design," *SIGMOBILE Mobile Computing Communication Review*, vol. 12, no. 1, pp. 56–58, 2008.

[16] Z. Miljanic, I. Seskar, K. Le, and D. Raychaudhuri, "The WINLAB Network Centric Cognitive Radio Hardware Platform - WiNC2R," *Mobile Network Applications*, no. 13, pp. 533–541, 2008.

[17] T. Ulversoy, "Software Defined Radio: Challenges and Opportunities," *IEEE Communications Survey and Tutorials*, vol. 12, no. 4, 2010.

[18] A. Jow, C. Schurgers, and D. Palmer, "CalRadio: a portable flexible 802.11 wireless research platform," in *Proceedings of the 1st International Workshop on System Evaluation for Mobile Platforms*, (San Juan, Puerto Rico), 2007.

[19] R. Manfrin, A. Zanella, and M. Zorzi, "Funcational and Perrformance Analysis of CalRadio 1 Platform," in *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications*, (Cambridge, MA, USA), 2009.

[20] K. Tan *et al.*, "Sora: High Performance Software Radio Using General Purpose Multi-core Processors," in *Proceedings of the Symposium on Networked Systems Design and Implementation*, (Boston, MA, USA), 2009.

[21] D. Lau, J. Blackburn, and J. Seely, "The Use of Hardware Acceleration in SDR Waveforms," in *Proceedings of the Software Defined Radio Technical Conference and Product Exposition*, (Orlando, FL, USA), 2005.

[22] V. Derudder *et al.*, "A 200 Mbps+ 2.14nJ/b digital baseband multi processor system-on-chip for SDRs," in *Proceedings of the Symposia on VLSI Technology and Circuits*, (Kyoto, Japan), 2009.

[23] U. Ramacher, "Software-defined radio prospects for multistandard mobile phones," *IEEE Computing Magazine*, vol. 40, no. 10, pp. 62–69, 2007.

[24] F. Clermidy, R. Lemaire, X. Popon, D. Ktenas, and Y. Thonnart, "An Open and Reconfigurable Platform for 4G Telecommunication: Concepts and Application," in *Proceedings of the Euromicro Conference on Digitial System Design*, (Patras, Greece), 2009.

[25] P. Agrawal, K. Sugand, M. Palkovic, P. Raghavan, L. V. der Perre, and F. Catthoor, "Partitioning and Assignment Exploration for Multiple Modes of IEEE 802.11n Modem on Heterogeneous MPSoC Platforms," in *Proceedings of the 15th Euromicro Conference on Digital System Design*, (Cesme, Izmir, Turkey), 2012.

[26] C. Silvano *et al.*, "Parallel Paradigms and Run-time Management Techniques for Many-core Architectures: The 2PARMA Approach," in *Proceedings of the IEEE 9th International Conference on Industrial Informatics*, (Caparica, Lisbon, Portugal), 2011.

[27] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[28] S. K. Singh, G. Singh, V. Pathak, and K. C. Roy, "Spectrum Management for Cognitive Radio based on Genetics Algorithm," *CoRR*, vol. abs/1101.4445, 2011.

[29] T. W. Rondeau, B. Le, C. J. Rieser, and C. W. Bostian, "Cognitive Radios with Genetic Algorithms: Intelligent Control of Software Defined Radios," in *Proceedings of the SDR Technical Conference and Product Exposition*, (Washington, DC, USA), 2004.

[30] C. Rieser, T. Rondeau, C. Bostian, and T. Gallagher, "Cognitive radio testbed: further details and testing of a distributed genetic algorithm based cognitive engine for programmable radios," in *Proceedings of the IEEE Military Communications Conference*, (Monterey, CA, USA), 2004.

[31] J. F. Hauris, "Genetic Algorithm Optimization in a Cognitive Radio for Autonomous Vehicle Communications," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, (Jacksonville, FL, USA), 2007.

[32] S. Bhattacharjee, A. Konar, and A. K. Nagar, "Channel Allocation for a Single Cell Cognitive Radio Network Using Genetic Algorithm," in *Proceedings of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, (Seoul, Korea), 2011.

[33] T. R. Newman, B. A. Barker, A. M. Wyglinski, A. Agah, J. B. Evans, and G. J. Minden, "Cognitive Engine Implmentation for Wireless Multicarrier Transceivers," *Wiley Journal on Wireless Communications and Mobile Computing - Cognitive Radio, Software Defined Radio and Adaptive Wireless Systems*, vol. 7, no. 9, pp. 1129–1142, 2007.

[34] C. B. Pettery, M. R. Leuze, and J. J. Grefenstette, "A Parallel Genetic Algorithm," in *Proceedings of the International Conference on Genetic Algorithm, Genetic Algorithms and Their Application*, (Cambridge, MA, USA), 1987.

[35] K. Twardowski, "An Associative Architecture for Genetic Algorithm-Based Machine Learning," *Computer*, vol. 27, no. 11, pp. 27–38, 1994.

[36] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[37] A. De Domenico, E. Strinati, and M.-G. Di Benedetto, "A Survey on MAC Strategies for Cognitive Radio Networks," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 1, pp. 21–44, 2012.

[38] E. Bonabeau and C. Meyer, "Swarm Intelligence: A Whole New Way to Think About Business," *Harward Business Review*, May 2001.

[39] C. Doerr, D. C. Sicker, and D. Grunwald, "Dynamic Control Channel Assignment in Cognitive Radio Networks using Swarm Intelligence," in *Proceedings of the IEEE Global Communications Conference*, (New Orleans, LA, USA), 2008.

[40] S. K. Udgata, K. P. Kumar, and S. L. Sabat, "Swarm Intelligence based Resource Allocation Algorithm for Cognitive Radio Network," in *Proceedings of the International Conference on Parallel, Distributed and Grid Computing*, (Waknaghat, Solan, India), 2010.

[41] B. Zhang, K. Hu, and Y. Zhu, "Spectrum Allocation in Cognitive Radio Networks Using Swarm Intelligence," in *Proceedings of the 2nd International Conference on Communication Software and Networks*, (Singapore), 2010.

[42] B. Atakan and O. B. Akan, "BIOlogically-inspired Spectrum Sharing in Cognitive Radio Networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, (Hong Kong SAR, PRC), 2007.

[43] G. Li, S. W. Oh, K. C. Teh, and K. H. Li, "Enhanced BIOlogically-inspired Spectrum Sharing for Cognitive Radio Networks," in *Proceedings of the IEEE International Conference on Communication Systems*, (Singapore), 2010.